Comunidades independientes de fútbol

A ver si me ayudas llevo mucho tiempo mirando esta página https://www.facebook.com/profile.php?id=100063611199563 y es muy larga y veo que hay muchísimos equipos de comunidades y selecciones que no conocía, pero como no doy abasto quiero que me digas como puedo tener una lista de todos los equipos mencionados (son de equipos de fútbol independiente a la FIFA)

Buscando perfiles y páginas de Facebook relacionadas

Encontré una página de Facebook "Federaciones Independientes" que parece tener información útil, pero el contenido es extenso y dinámico. Como no puedo acceder directamente al contenido completo, sugeriría que el usuario lo copie y pegue, o que utilice una extensión del navegador o un script en Python para extraer los nombres de los equipos.

Abriendo resultados de búsqueda con ID de referencia

Parece que no puedo acceder completamente a la página debido a restricciones de bloqueo. Puedo sugerir varias soluciones: usar "Buscar" (Ctrl+F) en el navegador, guardar la página y usar un script de Python para extraer nombres, o emplear extensiones como "SelectorGadget". Incluyo opciones según el nivel técnico del usuario, desde manual hasta herramientas avanzadas como Octoparse.

Puedo ofrecer un guion de Python que extrae nombres de equipos de HTML o texto plano usando expresiones regulares, como palabras en mayúsculas o nombres de países/territorios. El usuario deberá ejecutar el código por su cuenta, así que les compartiré el guion con instrucciones claras sobre cómo guardar la página y enviarme el contenido o PDF para procesarlo.

Listo

Perfecto — te explico rápido y con pasos concretos cómo conseguir una **lista de todos los equipos** que aparecen en esa página de Facebook (la que pasaste: `100063611199563` — “Federaciones Independientes”). Intenté abrir la página automáticamente pero Facebook bloquea el acceso por scraping, así que no puedo “bajarlo” yo mismo desde aquí. No pasa nada: te doy varias rutas —desde la más fácil a la más automática— y te dejo además un **script listo** que podés ejecutar en tu PC si querés una extracción automática y ordenada.

Opciones (elige la que prefieras)

Opción A — Rápida y manual (menos técnica)

1. Abrí la página en tu navegador (preferible en la versión de escritorio o móvil `m.facebook.com`).
2. Desplazate hacia abajo hasta cargar todo lo que quieras analizar (ir a pie por la página hasta donde llegues).
3. Seleccioná todo (Ctrl+A) y copialo (Ctrl+C).
4. Pegalo en un archivo de texto (Bloc de notas / Notepad / TextEdit) y guardalo como `pagina.txt`.
5. Si querés, pegalo aquí en el chat (o adjuntalo) y yo te ayudo a procesarlo y extraer los nombres.

Ventaja: simple; desventaja: podes necesitar mucho scroll y limpiar basura (comentarios, fechas, etc).

Opción B — Guardar la página y yo proceso el archivo

1. En el navegador: `Archivo → Guardar como...` → “Página web, completa” (o imprimir a PDF).
2. Subí ese archivo aquí (si podés adjuntar) o pegá el HTML/texto en el chat.
3. Yo lo proceso y te devuelvo la lista de equipos ordenada y única.

(Es la forma más directa si querés que lo haga yo.)

Opción C — Usar una extensión de navegador (automática sin programar)

Usá una extensión como **Web Scraper** (Chrome) o **Scraper / Simple HTML DOM**:

1. Instalar la extensión Web Scraper.
2. Abrir la página del perfil, crear un sitemap que capture los posts (selector: `.story_body_container` u otro; cada post).
3. Ejecutar la extracción y exportar como CSV.
4. Abrir el CSV y filtrar por nombres propios (o pegalo aquí y lo limpio).

Te paso pasos más detallados si te interesa esta ruta y decís qué navegador usás.

Opción D — (Recomendada si querés exactitud) Ejecutar un script Python local

Te dejo un script Python que:

- Toma un archivo HTML o texto (lo guardás después de copiar la página),
- Busca secuencias que parecen nombres de equipos (heurística: palabras con mayúscula, banderas emoji, palabras separadas por guiones, etc.),
- Limpia resultados comunes basura y te devuelve una lista única ordenada.

Copia esto a un archivo `extraer_equipos.py` y corré:

```
# extraer_equipos.py
# Uso: python extraer_equipos.py pagina.html > equipos.txt

import sys
import re
from html import unescape

if len(sys.argv) < 2:
    print("Usage: python extraer_equipos.py pagina.html")
    sys.exit(1)

with open(sys.argv[1], encoding="utf-8", errors="ignore") as f:
    txt = f.read()

# quitar tags html (simple)
txt = re.sub(r'<script.*?>.*?</script>', ' ', txt, flags=re.S|re.I)
txt = re.sub(r'<style.*?>.*?</style>', ' ', txt, flags=re.S|re.I)
txt = re.sub(r'<[^>]+>', ' ', txt)
txt = unescape(txt)

# Normalizar espacios
txt = re.sub(r'\s+', ' ', txt)

# Regla heurística 1: secuencias de 2-5 palabras que empiezan por mayúscula o contienen
banderas (□□) o símbolos
pattern1 = re.compile(r'([A-ZÁÉÍÓÚÑ][\wÁÉÍÓÚÑáéíóúñ&\.\-]{1,}\s(?:[A-ZÁÉÍÓÚÑ][\w\.\-]{1,}\s?)
{0,4})')
candidates = pattern1.findall(txt)

# Regla adicional: nombres con "/" o "-" (ej: 'Aymara - Mapuche', 'Team/Club')
pattern2 = re.compile(r'([A-ZÁÉÍÓÚÑ][\wÁÉÍÓÚÑáéíóúñ\-/&]{2,}(?:\s?[A-ZÁÉÍÓÚÑ][\w\.\-]{1,})*)')
candidates += pattern2.findall(txt)

# Incluir tokens con emoji bandera o 'vs' contextos
pattern_flag = re.compile(r'\b([A-Z][\w\-\s]{2,}?(?:□□|□□|□□|□□|□☠|□□|□))\b')
candidates += pattern_flag.findall(txt)

# Filtrar y limpiar
stopwords = set([
    'Federaciones','Fotos','Video','Departamento','Región','Campeonato','PARTIDO',
    'Full','Time','Half','Instagram','Facebook','Posted','Post','Publicado',
    'Home','Comments','Like','Siguenos','Grupo','Mujeres','Femenino','Masculino'
```

```
])
clean = []
for c in candidates:
    c = c.strip(" .,-:—_")
    if len(c) < 3 or len(c) > 60:
        continue
    # eliminar si contiene muchas minúsculas o números puros
    if sum(1 for ch in c if ch.isupper()) == 0:
        continue
    low = c.lower()
    if any(sw.lower() in low for sw in stopwords):
        continue
    # eliminar coincidencias obvias no nombres
    if re.match(r'^\d', c):
        continue
    clean.append(c)

# Unificar, ordenar, eliminar duplicados cercanos
uniq = []
seen = set()
for name in clean:
    n = re.sub(r'\s+', ' ', name).strip()
    nkey = n.lower()
    if nkey not in seen:
        seen.add(nkey)
        uniq.append(n)

uniq_sorted = sorted(uniq, key=lambda s: s.lower())

for u in uniq_sorted:
    print(u)
```

Cómo usarlo:

1. Abrí la página Facebook, hacé scroll hasta donde quieras.
2. Guardala como HTML (Ctrl+S) o copiá todo y pegá en `pagina.html`.
3. En tu PC, tenés Python (3.x). En terminal/cmd: `python extraer_equipos.py pagina.html > equipos.txt`
4. `equipos.txt` tendrá una lista. Pégala acá si querés que la revise y la normalice (por ejemplo, unir variantes, quitar errores).

Consejos para mejorar la precisión

- **Más contexto**: si pegás un bloque representativo de posts (unas 20-50 publicaciones), puedo probar el script aquí mismo y ajustar las reglas.
- **Filtrar por emoji bandera**: muchas entradas independientes usan banderas; esas ayudan a identificar selecciones.
- **Normalizar nombres**: a menudo aparecen variantes (ej. "Diaguita FC" vs "Diaguita"). Yo puedo agrupar variantes si me pasás la lista.
- **Excluir países/ciudades** que no sean equipos (p. ej. “Argentina”, “Chile”) — se pueden filtrar con una lista de stopwords.

Qué puedo hacer por vos ahora

- Si querés, pegá aquí un **trozo** (por ejemplo, copia de 10–20 posts) y lo proceso para darte la lista ya limpia.
- O subí el `pagina.html` o PDF y lo proceso yo.
- Si preferís que te guíe paso a paso para instalar la extensión o ejecutar el script en tu PC, te lo explico (Windows/Mac).

Decime cómo querés seguir y me pongo a trabajar en eso ahora mismo. ¿Querés pegar un fragmento o querés el script y lo ejecutás vos?

porque quiero, si es posible, bajarme el texto, las fotos y los comentarios de todos los usuarios (el tema es que pasan 10 minutos y ya suben algo nuevo=

ChatGPT dijo:
Pensado durante 14s

Para obtener información actualizada sobre las legalidades y métodos de consulta, el uso de ciertos servicios podría ser necesario. Si se trata de una página pública de Facebook, el contenido es accesible, pero el rasparlo puede violar términos. Para investigar, se podrá usar la API de Facebook, aunque se necesita un token de acceso adecuado.

Se puede usar la API de CrowdTangle, aunque está limitada para periodistas y investigadores. Si no hay acceso, se puede descargar la página manualmente como HTML o PDF, pero las imágenes pueden estar bloqueadas. En casos con muchos posts, se puede explorar el uso de herramientas de scraping como Selenium, aunque es importante advertir sobre las posibles violaciones de términos de servicio y bloqueos.

Si eres administrador de la página, puedes acceder a los datos desde "Configuración" y descargar la información o utilizar la API de Graph con un token de acceso. Si no eres administrador, puedes usar el acceso a contenido público a través de la API de Graph, aunque esto requiere permisos y revisión de la app. También se puede usar CrowdTangle si cumples los requisitos.

Puedes usar extensiones como SingleFile o "Save Page WE" para guardar páginas completas, o herramientas para descargar fotos en masa. También se puede utilizar un navegador sin cabeza y un script de Selenium para desplazarte por la página, guardar el HTML, y extraer imágenes y comentarios. Sin embargo, debes tener cuidado con las posibles restricciones y situaciones legales. Además, los posts nuevos cada 10 minutos se pueden gestionar mediante un trabajo programado o la API de Graph si eres administrador de la página.

Listo

Sí — se puede, **pero con muchas salvedades** legales y técnicas. Te doy un plan práctico (métodos reales que podés usar ahora), ejemplos de código y advertencias claras sobre privacidad y las políticas de Meta. Me tomo la libertad de proponerte 4 vías —de la más segura/legal a la más técnica— y te doy también un script ejemplo y cómo automatizarlo si querés.

Antes de arrancar — advertencias importantes

1. Si **no** sos administrador de la página, descargar en masa fotos y comentarios de **otros usuarios** puede violar los Términos de Meta y las leyes de privacidad en algunos países. Meta exige reglas específicas para la recolección automatizada y control de datos. Revisa las condiciones y permisos antes de hacerlo. Facebook
2. Si eres **administrador** de la página, tenés opciones mucho más directas y seguras (API de Página / exportación). Desarrolladores de Facebook+1
3. Para páginas públicas hay APIs (Graph API / Page Public Content Access) pero suelen requerir revisión y permisos; no es “abrir y listo”. Desarrolladores de Facebook
4. Para investigación periodística hay herramientas diseñadas (p. ej. CrowdTangle) pero el acceso también está restringido. GitLab

---

Opción 1 — Legal y robusta (si sos admin o tenés permiso)

- Si sos **administrador** de la página: usá la **Pages API** (Graph API) con un *page access token*. Podés listar posts, attachments, fotos, y comentarios vía endpoints `/PAGE_ID/posts`, `/POST_ID/comments`, `/POST_ID?fields=attachments` etc. Es la forma más ordenada y eficiente. Requiere crear una app en developers.facebook.com y posiblemente pasar revisión para permisos ampliados. Desarrolladores de Facebook+1

Qué hace falta (resumen):

1. Crear app en Meta for Developers.
2. Pedir permisos y review (si tu app necesita Page Public Content Access para páginas que no administrás).
3. Usar token de página para recorrer posts y descargar attachments (imagenes/videos) y comentarios.

Opción 2 — Herramientas para periodistas / investigadores

- **CrowdTangle** permite acceder a datos de páginas públicas de FB/IG para investigación (muy usado por periodistas). Requiere solicitud de acceso y suele aprobarse a medios, universidades o investigadores. Si te interesa el historial y monitorización de posts, consultá CrowdTangle. GitLab

---

Opción 3 — Guardado manual y extensiones (rápido, sin programar)

Ideal si no querés lidiar con APIs ni con revisar apps.

A) **SingleFile** — extensión para guardar una página completa en un solo HTML (incluye imágenes).

- Abrís la página, hacés scroll hasta cargar lo que quieras, clic en SingleFile → guarda todo como un HTML offline. Muy útil para capturar bloques de posts. SingleFile

B) **Extensiones de descarga de fotos** — hay extensiones que permiten descargar álbumes o imágenes visibles en pantalla (no oficiales). Útil si querés bajar fotos del feed o álbumes puntuales; ojo con limitaciones y privacidad. Chrome Web Store

Ventaja: rápido y sin programar.
Desventaja: no escalable, y si hay posts nuevos cada 10 minutos deberías repetir el proceso.

Opción 4 — Automatizar con un navegador controlado (Selenium / Puppeteer)

Si querés **extraer continuamente** (cada X minutos) y no sos admin, la técnica común es usar un navegador automatizado que:

- inicia sesión (si es necesario),
- hace scroll infinito para cargar posts,
- extrae html/posts, descarga imágenes y guarda comentarios.

Importante: esto técnicamente funciona pero con riesgos: bloqueo de cuenta/IP, cambios de diseño que rompen el script y problemas legales/ToS. Usalo solo para páginas públicas y con cuidado. Referencias sobre scroll/automatización: guías de Selenium. BrowserStackStack Overflow

A continuación te dejo un **ejemplo** (Python + Selenium + BeautifulSoup) que hace scroll y guarda HTML y descarga imágenes encontradas. Es un punto de partida — tenés que adaptarlo y poner límites para no sobrecargar la página.

```
# ejemplo_scrape_facebook.py  (ejemplo educativo)
# Requisitos: pip install selenium beautifulsoup4 requests
# Necesitas chromedriver compatible con tu Chrome y ajustar PATH.
```

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
import time, os, requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin, urlparse

URL = "https://www.facebook.com/profile.php?id=100063611199563"
OUTDIR = "fb_dump"
SCROLL_PAUSES = 2
MAX_SCROLLS = 50  # bajar si querés poco

os.makedirs(OUTDIR, exist_ok=True)

opts = Options()
opts.add_argument("--start-maximized")
# opts.add_argument("--headless=new")  # no recomendado si necesitas cargar JS dinámico
driver = webdriver.Chrome(options=opts)

try:
    driver.get(URL)
    time.sleep(5)  # esperar carga inicial

    last_height = driver.execute_script("return document.body.scrollHeight")
    for i in range(MAX_SCROLLS):
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(SCROLL_PAUSES)
        new_height = driver.execute_script("return document.body.scrollHeight")
        if new_height == last_height:
            break
        last_height = new_height
        print("scroll", i+1)

    html = driver.page_source
    with open(os.path.join(OUTDIR, "page.html"), "w", encoding="utf-8") as f:
        f.write(html)

    soup = BeautifulSoup(html, "html.parser")

    # ejemplo simple: bajar imágenes <img>
    imgs = soup.find_all("img")
    count = 0
    for img in imgs:
        src = img.get("src")
        if not src:
            continue
        # normalizar urls
        parsed = urlparse(src)
        if not parsed.scheme:
            src = urljoin(URL, src)
        try:
            r = requests.get(src, timeout=15)
            if r.status_code == 200:
                fn = f"img_{count}.jpg"
                with open(os.path.join(OUTDIR, fn), "wb") as f:
                    f.write(r.content)
                count += 1
        except Exception as e:
            print("err img", e)

    print("Guardado HTML y", count, "imágenes en", OUTDIR)

finally:
    driver.quit()
```

Cómo automatizar periódicamente (ejemplo en Linux): poner este script en un cron que corra cada 10 minutos:

```
*/10 * * * * /usr/bin/python3 /ruta/a/ejemplo_scrape_facebook.py
```

Pero **no** lo dejes correr sin límites: respeta pausas, evita hacer muchas peticiones simultáneas y revisá la legalidad.

Sugerencia para tu caso (posts cada 10 min)

- Si **sos admin**: usar la Graph API y programar un job que consulte `/PAGE_ID/posts?since=TIMESTAMP` cada 5–10 min y baja sólo lo nuevo. Es limpio y no te bloquean. Desarrolladores de Facebook
- Si **no** sos admin: podes programar la opción Selenium pero con muchos cortafuegos: limitar frecuencia, rotar IPs con cuidado, y preferir descargar solo “nuevos” posts comparando IDs, para no repetir. **O mejor** pedir al administrador que te facilite los archivos o permisos (es lo más seguro).

Si querés, ahora mismo puedo:

1. Enviarte un script más avanzado que además:
    - Extraiga la estructura de posts (texto), autor, fecha, enlace del post, IDs de comentario y texto de comentarios;
    - Normalice nombres de equipos y produzca un CSV.
2. O guiarte paso a paso para instalar SingleFile o una extensión de fotos y enseñarte a guardar automáticamente (por ejemplo: usar AutoHotkey o un programador de tareas para abrir la página y guardar con la extensión).

Decime cuál de estas 2 opciones preferís y te doy el script final listo o la guía paso a paso para tu navegador/sistema (Windows o Linux). También puedo generarte el script para monitorizar sólo *lo nuevo* y que lo guarde en un CSV.

no soy admin, apenas le di me gusta, y siguiendo, intenté copiar todo pero es super tedioso y no se termina mas

ChatGPT dijo:
Pensado durante 6s

El usuario quiere una solución práctica para obtener actualizaciones de una página pública de Facebook, específicamente publicaciones, fotos y comentarios. Aunque no podemos ayudar con métodos que violen los términos de servicio, sugeriré opciones legales como la función de guardar en Facebook o usar extensiones como SingleFile con automatización para realizar descargas periódicas. También se podría considerar usar Selenium, pero con precauciones sobre el bloqueo de la cuenta.

Para obtener actualizaciones periódicas de una página de Facebook, sugiero dos opciones:

1. Usar la extensión SingleFile junto con un script que haga scroll automáticamente y guarde la página cada cierto tiempo.
2. Utilizar una herramienta como 4K Stogram o una extensión como DownAlbum para Facebook, aunque éstas pueden ser más limitadas.

Para obtener actualizaciones periódicas, sugiero usar un script en Puppeteer o Python Selenium. El script puede detectar nuevas publicaciones mediante post ID, guardar los datos en formato JSON y descargar imágenes. Para mayor seguridad, recuerda cumplir con los términos de servicio y considera pedir permiso al administrador de la página.

Para extraer publicaciones de Facebook, una opción es buscar los enlaces con "posts" en el atributo `href` de las etiquetas `<a>` relacionadas con la marca de tiempo. Usando BeautifulSoup, podemos analizar la página y obtener los IDs de las publicaciones. Guardamos estos IDs en un archivo JSON y solo descargamos los nuevos, incluyendo imágenes.

Listo

Perfecto — te doy **dos soluciones concretas** (una sin programar y otra automatizada) que podés poner en marcha ahora mismo. Ambas están pensadas sabiendo que **no sos admin** (o sea: hacelo con cuidado y para uso personal; revisá siempre la legalidad/ToS). Te explico paso a paso y te dejo un **script listo** (Python + Selenium) que hace lo siguiente automáticamente cada vez que lo corrés: hace scroll, detecta posts nuevos (por ID), abre cada post nuevo, expande comentarios, guarda el texto y descarga las fotos.

Antes de todo, un recordatorio importante y claro:
Advertencia legal / de riesgo — Automatizar descargas masivas de contenido de Facebook puede violar los Términos de Meta y puede conllevar bloqueos de cuenta o IP. Usá esto sólo para páginas públicas y para uso personal/legítimo; lo más aconsejable es pedir permiso al administrador de la página.

Opción A — Sin programar (la más simple para empezar)

Si no querés instalar nada, esta opción es la más segura a corto plazo:

1. Instala la extensión **SingleFile** (Chrome/Firefox).
2. Abre la página de Facebook, empieza a hacer scroll manual hasta donde quieras (o dejá el navegador en automático con un script de autoscroll muy simple).
3. Cada 15–30 minutos: clic en SingleFile → “Guardar página” → elegí un nombre con fecha/hora (`federaciones_2025-09-01_2100.html`). Guarda muchas copias con timestamp.
4. Cuando tengas varios snapshots, me podés mandar uno (o varios) y yo extraigo la lista de equipos, textos y metadatos.

Ventajas: no programar, sencillo. Desventajas: manual si hay muchas actualizaciones y no descarga comentarios que estén colapsados hasta que los expandas.

Para automatizar ese “cada X minutos” sin programar: podés usar AutoHotkey (Windows) o una acción programada del sistema que abra Chrome, haga scroll (AutoHotkey) y active SingleFile. Si querés, te doy el AutoHotkey listo.

Opción B — Automatizada (recomendada si querés captura periódica cada 10 min)

Requiere instalar Python, Chrome y ChromeDriver. Te doy un script completo que:

- hace scroll,
- detecta **IDs** de post (para sólo bajar lo nuevo),
- abre cada post nuevo para extraer texto y comentarios (expandir “ver más comentarios”),
- descarga imágenes del post,
- guarda todo en JSON y carpetas por post,
- mantiene un archivo `seen_posts.json` para no repetir.

Instalación rápida (Windows / Linux):

1. Tenés que tener Python 3.x instalado.
2. `pip install selenium beautifulsoup4 requests`
3. Descargá ChromeDriver que coincida con tu Chrome y ponelo en tu PATH o en la misma carpeta del script.
4. Copiá este script a `fb_monitor.py`.

```
# fb_monitor.py
# Uso: python fb_monitor.py
# Requisitos: pip install selenium beautifulsoup4 requests

import os, time, json, random, re
from pathlib import Path
```

```
from urllib.parse import urljoin, urlparse
import requests
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By

URL = "https://www.facebook.com/profile.php?id=100063611199563"
OUTDIR = Path("fb_dump")
SEEN_FILE = OUTDIR / "seen_posts.json"
MAX_SCROLLS = 30
SCROLL_PAUSE = 2.0

OUTDIR.mkdir(parents=True, exist_ok=True)
if not SEEN_FILE.exists():
    SEEN_FILE.write_text("[]", encoding="utf-8")

seen = set(json.loads(SEEN_FILE.read_text(encoding="utf-8")))

opts = Options()
# para ver lo que pasa, no headless; si querés headless, comentá siguiente línea
# opts.add_argument("--headless=new")
opts.add_argument("--start-maximized")
driver = webdriver.Chrome(options=opts)

def scroll_to_bottom(max_scrolls=MAX_SCROLLS):
    last_h = driver.execute_script("return document.body.scrollHeight")
    for i in range(max_scrolls):
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(SCROLL_PAUSE + random.uniform(0.5,1.5))
        new_h = driver.execute_script("return document.body.scrollHeight")
        if new_h == last_h:
            break
        last_h = new_h

def find_post_links(html):
    soup = BeautifulSoup(html, "html.parser")
    links = set()
    # buscar enlaces típicos de posts/permalinks
    for a in soup.find_all("a", href=True):
        h = a["href"]
        if "permalink.php" in h or "/posts/" in h or "story_fbid" in h:
            # normalizar a URL completa si es relativo
            if h.startswith("/"):
                h = urljoin("https://www.facebook.com", h)
            links.add(h.split('?')[0])
    # también buscar data-ft u otros ids
    # extraer posibles fbid con regex en el html
    for m in re.finditer(r'story_fbid[^\d]*(\d+)', html):
        links.add("https://www.facebook.com/permalink.php?story_fbid=" + m.group(1))
    return sorted(links)

def get_post_id_from_url(u):
    m = re.search(r'(?:permalink\.php\?story_fbid=|/posts/)(\d+)', u)
    return m.group(1) if m else None

def download_url(url, outpath):
    try:
        r = requests.get(url, timeout=20)
        if r.status_code == 200:
            outpath.write_bytes(r.content)
            return True
    except Exception as e:
        print("download err", e)
    return False

try:
    driver.get(URL)
    time.sleep(5 + random.uniform(0.5,1.5))
    scroll_to_bottom()
```

```
    html = driver.page_source
    post_links = find_post_links(html)
    print("Encontrados enlaces (candidatos):", len(post_links))

    new_posts = []
    for pl in post_links:
        pid = get_post_id_from_url(pl)
        if not pid:
            continue
        if pid in seen:
            continue
        print("Nuevo post:", pid, pl)
        new_posts.append((pid, pl))

    for pid, pl in new_posts:
        # abrir en nueva pestaña para aislar
        driver.execute_script("window.open(arguments[0], '_blank');", pl)
        driver.switch_to.window(driver.window_handles[-1])
        time.sleep(3 + random.uniform(0.5,1.5))

        # intentar expandir "Ver más comentarios" y "Ver más"
        for _ in range(6):
            try:
                # botones de "ver más comentarios" o "Ver más"
                btns = driver.find_elements(By.XPATH, "//div[contains(text(),'Ver más
comentarios') or contains(text(),'View more comments') or contains(text(),'Ver más')]")
                if not btns:
                    break
                for b in btns:
                    try:
                        driver.execute_script("arguments[0].click();", b)
                        time.sleep(1+random.random())
                    except:
                        pass
            except:
                break

        post_html = driver.page_source
        soup = BeautifulSoup(post_html, "html.parser")

        # Extraer texto del post (heurística)
        post_text = ""
        # muchos posts tienen aria-label o divs con role=article
        article = soup.find(attrs={"role":"article"})
        if article:
            post_text = article.get_text(" ", strip=True)
        else:
            post_text = soup.get_text(" ", strip=True)[:5000]

        post_dir = OUTDIR / pid
        post_dir.mkdir(exist_ok=True)

        # guardar JSON con metadatos y texto
        meta = {"post_id": pid, "url": pl, "text": post_text, "images": [], "comments": []}
        # extraer imágenes
        for img in soup.find_all("img", src=True):
            src = img["src"]
            if any(x in src for x in ["scontent","fbcdn",".jpg",".png",".jpeg"]):
                fname = src.split("?")[0].split("/")[-1]
                outp = post_dir / fname
                if download_url(src, outp):
                    meta["images"].append(str(outp))
        # extraer comentarios (heurística: buscar bloques con aria-label "Comentario" o role
listitem dentro article)
        if article:
            for li in article.find_all(["div","span"], recursive=True):
                # heurística simple: líneas con varios tokens y nombres en mayúscula inicial
                t = li.get_text(" ", strip=True)
                if t and len(t) > 20 and any(ch.isalpha() for ch in t):
```

```
                    # ignorar texto demasiado grande (probablemente todo el post)
                    if len(t) < 500:
                        meta["comments"].append(t)
        # guardar JSON
        with open(post_dir / "meta.json", "w", encoding="utf-8") as jf:
            json.dump(meta, jf, ensure_ascii=False, indent=2)

        seen.add(pid)
        # cerrar pestaña
        driver.close()
        driver.switch_to.window(driver.window_handles[0])
        time.sleep(2 + random.random())

    # actualizar seen file
    with open(SEEN_FILE, "w", encoding="utf-8") as f:
        json.dump(sorted(list(seen)), f, ensure_ascii=False, indent=2)

    print("Proceso terminado. Nuevos posts procesados:", len(new_posts))

finally:
    driver.quit()
```

Cómo usarlo y automatizarlo:

- Ejecutá `python fb_monitor.py`. Verás carpetas `fb_dump/<post_id>/meta.json` y las imágenes.
- Para que corra cada 10 minutos:
  - En Windows: Programador de tareas → Crear tarea → Ejecutar `python C:\ruta\fb_monitor.py` cada 10 minutos.
  - En Linux: editar `crontab -e` y añadir `*/10 * * * * /usr/bin/python3 /ruta/fb_monitor.py`.

Consejos anti-bloqueo:

- No ejecutes consultas continuas a menos que haga scroll y espere (el script ya incluye pausas aleatorias).
- Evitá más de 1 ejecución cada 5–10 minutos si no sos admin.
- Si recibís bloqueos, frenalo y considera pedir permiso o usar snapshots manuales.

Cómo mejora la calidad de extracción (si querés que lo ajuste yo)

- Si me pegás aquí (o subís) **un HTML guardado** o un **snippet** de ~10 publicaciones, lo proceso y te devuelvo el CSV con: `post_id, fecha, autor (si aparece), texto, número_de_comentarios, urls_imagenes`.
- Si querés, te adapto el script para **solo bajar nombres de equipos** (limpieza, normalización y agrupación), o para ignorar posts con cierto texto.

Otras alternativas (si no querés programar)

- Extensiones como **DownAlbum** / **Image Downloader** para bajar imágenes del feed (rápidas pero menos ordenadas).
- Pedir al admin de la página que exporte o comparta los archivos (lo ideal, si podés contactarlo).
- Solicitar acceso a **CrowdTangle** (si tu caso es periodístico/investigativo) — pero eso requiere aprobación.

Decime qué preferís ahora:

1. Te doy el **AutoHotkey** para que SingleFile guarde la página cada X minutos automáticamente.
2. Probás el **script Python** y te explico cómo programarlo (te doy comandos para Windows Task Scheduler paso a paso).
3. Me pegás un **trozo de la página** (10–20 posts) y yo te extraigo la lista de equipos y comento cómo normalizarlos.

No preguntes: elegí una de las 3 opciones y te doy los pasos concretos listos para pegar. ¿Cuál querés?

Dijiste:

que pena que no la puedas ver https://www.facebook.com/profile.php?id=100063611199563 para que sepas lo completa que es, nunca vi nada igual, esto debería estar en la RFFFS de hecho ahí saqué el link

Parece que el usuario está comentando sobre un enlace y su admiración, pero no pide una acción concreta. Como no puedo acceder automáticamente a Facebook debido a las restricciones, ofreceré ayuda. Si puede proporcionar un fragmento o HTML, podría procesarlo. Además, puedo guiarlo si desea contactar a RFFFS o usar scripts locales.

Puedo ofrecerte un par de opciones para ayudarte: (1) Un script de AutoHotkey que guarda la página periódicamente, o (2) un script mejorado de Selenium para monitorear solo nuevas publicaciones. También, si lo prefieres, puedes pegar un fragmento y extraeré los equipos. ¡Elije lo que te resulte más fácil!

Voy a ofrecer las siguientes opciones: (1) Instrucciones para usar SingleFile con un intervalo recomendado de 10 minutos, y (2) un script de AutoHotkey. Además, recordaré brevemente la importancia de respetar las políticas de uso. También te guiaré paso a paso para configurar el atajo de teclado y usarlo. ¡Con gusto puedo ayudarte a procesar contenido pegado ahora!

SingleFile guarda páginas usando un atajo de teclado como Ctrl+Shift+S, que podemos configurar. El script de AutoHotkey abrirá Chrome, hará scroll y activará este atajo para guardar la página. Si SingleFile está configurado para guardar automáticamente, no aparecerá el cuadro de confirmación. Instrucciones claras para configurar la extensión y el script.

Listo

Genial — gracias por compartirlo, se nota que te interesa mucho y que la página está re completa. No puedo abrirla desde aquí (Facebook bloquea ese tipo de scraping remoto), pero te doy **dos soluciones prácticas e inmediatas** para que puedas empezar a capturar todo de forma cómoda — una **fácil, sin programar** (usa la función de “Guardar página”) y otra **automática con AutoHotkey** para Windows que te hará snapshots periódicos (cada X minutos) y les pondrá nombre con timestamp.

También, si preferís, **pegá aquí 10–20 publicaciones** (copiar/pegar) y yo ahora mismo te extraigo la lista de equipos y normalizo nombres. Decime cuál querés: AHK o que lo haga yo con un fragmento.

Opción rápida (sin programar)

1. Instala la extensión **SingleFile** o usa `Ctrl+S` para “Guardar como → Página web, completa”.
2. Abre la página del perfil, hace scroll hasta donde quieras (o dejá que cargue bastante).
3. Guardá la página con `Ctrl+S` y poné nombre con fecha/hora (ej. `federaciones_2025-09-01_2100.html`).
4. Repetí cada 10 min si querés capturar lo nuevo. (Si querés, más abajo te doy la automatización AHK para que lo haga por vos.)

Ventaja: muy simple. Desventaja: manual si hay muchas actualizaciones.

Opción automática (AutoHotkey — Windows)

Este script abre Chrome en la URL que indicas, hace scroll para cargar contenido, guarda la página con `Ctrl+S` y pone un nombre con fecha/hora. Repite cada X minutos (configurable). Funciona sin tocar APIs de Facebook — simplemente guarda snapshots HTML - eso suele ser suficiente para después extraer texto, imágenes y comentarios.

Pasos previos

1. Instala AutoHotkey: https://www.autohotkey.com/
2. Asegurate de que Chrome sea tu navegador por defecto o ajustá la ruta en el script si usás otro.
3. Crea un archivo `fb_autosave.ahk` y pegá el script siguiente.

Script (cópialo tal cual en fb_autosave.ahk):

```
; fb_autosave.ahk
; Ejecutar: doble clic en fb_autosave.ahk
; Editar URL y INTERVAL_MIN a tu gusto.

URL := "https://www.facebook.com/profile.php?id=100063611199563"
INTERVAL_MIN := 10  ; cada cuántos minutos guardar
SAVE_DIR := "C:\Users\%A_UserName%\Downloads\FB_SNAPSHOTS" ; carpeta donde se guardan los html

; crear carpeta si no existe
FileCreateDir, %SAVE_DIR%

Run, chrome.exe --new-window "%URL%"
WinWaitActive, ahk_exe chrome.exe, , 15
Sleep, 5000

Loop
{
    ; 1) hacer scroll hasta abajo lentamente para cargar posts
    Loop, 20
    {
        Send, {End}
        Sleep, 1200
        Send, {WheelDown 6}
        Sleep, 800
    }

    ; 2) Guardar página con Ctrl+S
    Send, ^s
    Sleep, 800

    ; 3) construir nombre con timestamp YYYYMMDD_HHMM
    FormatTime, timestamp, , yyyyMMdd_HHmm
    filename := "federaciones_" . timestamp . ".html"
    fullpath := SAVE_DIR . "\" . filename

    ; 4) Pegar ruta+nombre y pulsar Enter
    ; Nota: en algunos sistemas el cuadro "Guardar como" se abre distinto; este enfoque
funciona en la mayoría
    Send, %fullpath%
    Sleep, 300
    Send, {Enter}
    Sleep, 3000

    ; 5) esperar un poco para que termine la descarga
    Sleep, 4000

    ; 6) volver a la página principal (si se abrió tab nuevo)
    WinActivate, ahk_exe chrome.exe
    Sleep, 500

    ; esperar INTERVAL_MIN minutos antes de la próxima guardada
    Sleep, % INTERVAL_MIN * 60000
```

```
}
```

Notas / ajustes

- Cambiá `INTERVAL_MIN` a `10` (o lo que quieras).
- Cambiá `SAVE_DIR` si querés otra carpeta.
- Si Chrome pregunta confirmaciones de guardado, probá antes una ejecución manual para ver cómo responde tu sistema.
- Si preferís SingleFile (mejor para tener todo embebido en un solo `.html`), podés asignar un atajo de teclado a la extensión (en `chrome://extensions/shortcuts`) y cambiar la parte `Send, ^s` por `Send, ^+s` o la combinación que elijas.

Advertencia legal / de uso

- Hacé esto sólo para uso personal y páginas públicas. No automatices a lo loco (riesgo de bloqueo o de violar ToS).
- Evitá hacer demasiadas ejecuciones seguidas (el script usa intervalo por defecto de 10 minutos, prudente).

Qué puedo hacer ahora (elige una)

1. Te doy el **script Selenium** más pulido (si querés un JSON con comentarios e imágenes) y te explico cómo ejecutarlo en tu PC.
2. Te doy el **AutoHotkey** aún más avanzado (por ejemplo, que mueva archivos a carpetas por fecha, o que descargue sólo imágenes).
3. **Pegá aquí** un bloque de 10–20 publicaciones (texto) y **ahora mismo** te extraigo la lista de equipos y te la devuelvo limpia y ordenada.

Decime cuál opción preferís y te doy lo siguiente en el mismo mensaje (sin esperas): el archivo/config listo o yo proceso el texto que pegues. ¿Querés que empiece procesando 10–20 posts que pegues acá?